This application is submitted in the name of the following inventor(s):

Inventor Name	Citizenship	Residence City and State
Kleiman, Steven R.	United States	Los Altos, California

The assignee is Network Appliance, Inc., a corporation having an office at 495 Java Drive, Sunnyvale, California, 94089.

TITLE OF THE INVENTION

Communication of Control Information and Data in Client/Server Systems

BACKGROUND OF THE INVENTION

Field of the Invention

The invention relates to computer communication, such as in client/server systems.

Related Art

One known model for assigning or performing tasks in a computer system is a client/server model. In a client/server model, clients make requests for service (by sending messages) to a server; the server responds to those requests for service by providing services to requesting clients (and possibly sending messages back to requesting clients). For example, the server might include a file server responsive to file system requests, a web server responsive to network object requests, a database server responsive to database requests, or some other type of server. Client/server models are used both internally within a single device (the client and server are different software modules), as well as between different devices (the client and server are coupled by a communication link).

When the client and server are different devices, they communicate using a communication link. In byte serial systems, messages between devices are sent and received using a communication protocol. Each message has prepended header information indicating its intended recipient, payload information, and appended checksum information. The sender thus wraps the message inside a serial byte stream, which the receiver unwraps to determine what the message is. Often, the communication protocol will be multi-layered — a lower-level protocol carries multiple types of messages, while different higher-level protocols carry messages suited to particular purposes. Thus, higher-level protocol messages package communication between the

client and server, while lower-level protocol messages break up the higher-level protocol messages and package portions of it for sending between devices.

While byte serial models are designed for a very open and diverse environment, they are not well suited to rapid communication of relatively large blocks of data. First, relatively large blocks of data must generally be broken up by the sender into smaller messages, so as to accommodate the message sizes of intermediate communication links. Similarly, the smaller messages must be reassembled at the receiver into the relatively larger blocks of data; this is called fragmentation and reassembly. Second, payload information is not reliably located at any aligned location when received; this causes the receiver to move the payload information into a buffer where the block of data is aligned at a known position. Third, checksum information is computed by the sender and checked by the receiver for each message; this produces substantial computing overhead for each message and for each block of data. Fourth, the receiver must generally be prepared to receive messages of up to the largest possible size; this causes the receiver to allocate maximum size buffers, which are often larger than necessary.

Another known method for communicating data includes DMA (direct memory access) transfer of data between devices. One such method of DMA transfer is known as NUMA (non-uniform memory access); examples of NUMA architectures include Infiniband, ServerNet and interconnection networks compliant with the VI

(Virtual Interface) architecture standard such as cLan, Servernet II, and FC-VI. Using a DMA transfer, the initiating device transfers data directly to or from a memory for the target device. The specific memory locations on the target device are specified by the initiator using addresses associated with addresses on the target device. While NUMA architectures are well suited to rapid communication of relatively large blocks of data, they are not generally designed to support high latency wide area networks or to support networks in which export of memory is problematic for security reasons. NUMA architectures are best suited to communication between devices that are closely coupled, both using hardware (relatively short haul communication links) and software (relatively closely cooperating system elements).

One system has used NUMA architecture for communication in a client/server architecture. The Microsoft "Winsock Direct Path" sends messages between client and server using both a TCP/IP communication link and a NUMA communication link. The Winsock Direct Path architecture, after wrapping the message for communication between the sender and the receiver, determines if there is a NUMA communication link available; if so, the Winsock Direct Path architecture uses that NUMA communication link to send the message; if not, the Winsock Direct Path architecture uses the TCP/IP communication link. While the system has some of the advantages of communication using a NUMA architecture, it still has the drawbacks noted earlier for byte serial models of communication in a client/server architecture.

Accordingly, it would be advantageous to provide a technique involving computer communication systems, such as those using a client/server model, that is not subject to drawbacks of the known art.

SUMMARY OF THE INVENTION

The invention provides a method and system in which a client/server system uses a NUMA communication link, possibly in combination with a byte serial communication link, to transfer relatively large blocks of data between client and server. The method and system provides for transferring data between the client and server, in which timing for the actual data transfer is decoupled from a request (from the client) or a response (from the server). The method and system also provides for transferring data from either party to the other at mutually agreed locations, such as locations responsive to control information present in either the request or the response. Accordingly, either party can transfer data to the other at a location convenient to both the sender and the recipient, and either party can process data in any order it prefers, without regard for the order in which data is stored at the other party.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows a block diagram of a client/server system using a NUMA communication link.

Figure 2 shows a process flow diagram of a method of using a system as in figure 1.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

System Elements

Figure 1 shows a block diagram of a client/server system using a NUMA communication link.

A system 100 includes a server 110, a communication link 120, and one or more clients 130.

The server 110 includes a processor, server program and data memory 111, and server mass storage 112. The server memory 111 includes server software 113, including instructions for receiving requests from clients 130 and providing responses to clients 130, and server data buffers 114, including locations for receiving information from clients 130 and for recording information for access by clients 130.

The server data buffers 114 are matched to the size of data blocks to be transferred into or out of those server data buffers 114. Accordingly, a first set of server data buffers 114 are relatively larger (such as about 4 Kbytes), so as to accommodate

relatively larger data blocks such as disk blocks. A second set of server data buffers 114 are relatively smaller (such as about 256 bytes), so as to accommodate relatively smaller data blocks such as control information. As described in detail below, control information can include memory addresses (either at the server 110 or at the client 130), client/server requests or responses, status information, checksums, or other information for communication between client 130 and server 110 that is relatively smaller than a disk block.

Although this application describes the preferred embodiment as having one server 110, this description is for simplicity only. An embodiment of the system may include more than one server 110, which may each communicate with more than one client 130. The set of servers 110 serving a first client 130 can overlap with the set of servers serving a second client 130; similarly, the set of clients 130 being served by the first server 110 can overlap with the set of clients being served by a second server 110. Moreover, servers 110 can communicate with each other, and clients 130 can communicate with each other, including using techniques described herein with regard to client/server communication.

The communication link 120 includes one or more NUMA communication links 121. In an alternative embodiment, the communication link 120 might also include one or more byte serial communication links 122; however, these adjunct byte serial communication links 122 are not required.

The NUMA communication links 121 allow clients 130 and servers 110 to read or write directly into each other's memory 131 or memory 111, using DMA memory read and write operations. Thus, the server 110 can read or write directly into or out of client memory 131, or clients 130 can read or write directly into or out of server memory 111. There is no particular requirement regarding which locations in the client memory 131 or server memory the client 130 or server 110 can read or write directly into or out of. Target addresses may have to be explicitly exported before they are remotely accessible; however in a preferred embodiment, server 110 does not export memory.

In an alternative embodiment, the byte serial communication links 122 allow clients 130 and servers 110 to send and receive messages 140 to each other. As noted earlier, these adjunct byte serial communications links 122 are not required.

Similar to the server 110, each client 130 includes a processor, client program and data memory 131, and client mass storage 132. The client memory 131 includes client software 133, including instructions for presenting requests to the server 110 and receiving responses from server 110, and client data buffers 134, including locations for receiving information from server 110 and for recording information for access by the server 110.

Similar to the server 110, the client data buffers 134 are matched to the size of data blocks to be transferred into or out of those client data buffers 134. Accordingly,

a first set of client data buffers 134 are relatively larger (such as about 4 Kbytes), so as to accommodate relatively larger data blocks such as disk blocks. A second set of client data buffers 134 are relatively smaller (such as about 256 bytes), so as to accommodate relatively smaller data blocks such as control information. These sets of client data buffers 134 need not be the same size as those of the server 110. The sizes indicated are purely illustrative and in no way limiting.

Requests from the client 130 includes addresses within client buffer 134 where results of a read request or a write request should be directed from the server buffer 114.

Method of Use

A method 200 is performed by the system 100. Although the method 200 is described serially, the steps of the method 200 can be performed by separate elements in conjunction or in parallel, whether asynchronously, in a pipelined manner, or otherwise. Lastly, there is no particular requirement that the method 200 be performed in the same order in which this description lists the steps, except where so indicated.

At a flow point 200, the system 100 ready to begin performing a method 200. The server 110 and the client 130 are ready to send and receive messages.

Request from the Client

At a step 205, the client 130 exports or passes an address located within the client data buffer 134 to the NUMA communication link 121. This address allows computing resources to be used most efficiently because the server 110 can direct it's response to the request in such a way as to make optimal use of the space available in the client data buffer 134. The address of the client data buffer 134 is responsive to the relative size of the data block that will be transferred.

If the request is a read request, a client 130 might pass an address of the client data buffer 134 that should receive the results of a read. If the request is a write request, the client 130 might pass the specific address of the client data buffer 134 that should contain data to be written.

In a step 210, the address is transferred from the NUMA communication link 121 to the server 110.

Response of the Server

At a step 215, the server 110 receives the address of the client data buffer 134 and begins processing it.

At a step 220, the server 110 transfers data from one of the server data buffers 114 to a NUMA communication link 121. It should be noted that the actual data transfer is decoupled from the request of the client 130.

At a step 225, the data is transferred using the NUMA communication link 121 to the specified address in one of the client data buffers 134. If the client request was a read request, the data is transferred from the NUMA communication to the specified address of the client data buffer 134. If the client request was a write request, the server 110 reads the data located at the specified address at a client data buffer 134. In a preferred embodiment, the client data buffers 134 are of different sizes and alignments than the server data buffers 114.

The data transfer can be asynchronious; processing of data can occur in any order that is particularly convenient to the server 110 and client 130 as long as the transfer of the data is responsive to the request.

Alternative Embodiments

Although preferred embodiments are disclosed herein, many variations are possible which remain within the concept, scope, and spirit of the invention, and these variations would become clear to those skilled in the art after perusal of this application.

Generality of the Invention

The invention has general applicability to various fields of use, not necessarily related to e-commerce as described above. For example, these fields of use can include one or more of, or some combination of, the following:

- sending requests from a client device to a database server and transferring data from the database server to a client device in response to the request

- sending requests from a client device to a mail server and transferring data from a mail server to a client device in response to the request

- sending requests from a client device to a cache or proxy server and transferring data from a cache server to a client device in response to the request

- sending requests from a client device to a web server (HTTP) and transferring data from a web server (HTTP) to the client device in response to the request

- sending requests from a client device to an FFT server and transferring bulk data from the FFT server to the client device in response to the request.